International Journal of Control and Automation
Vol.7, No.6 (2014), pp.401-412
http://dx.doi.org/10.14257/ijca.2014.7.6.37

Reducing Energy Consumption in Wired OpenFlow-Based Networks

Bhed Bahadur Bista[1], Arata Fukushi[1], Toyoo Takata[1] and Danda B. Rawat[2]

[1]Faculty of Software and Information Science
Iwate Prefectural University, Iwate, Japan, 020-0693
[2]Georgia Southern University, Statesboro, Georgia 30460, USA

{bbb, takata}@iwate-pu.ac.jp, db.rawat@ieee.org

Abstract

Due to the proliferation of mobile devices such as smart phones and tablet computers, and increase in the speed of network devices to forward data, the Internet traffic has increased manyfold compare to a decade ago. Energy consumed by network devices handling the network traffic has increased manyfold also. Researchers are not only focusing on battery oriented network such as wireless sensor and ad hoc networks on saving battery power to extend network life but are also focusing on wired network to save energy consumed by network devices. Energy consumed by wired network can be reduced by putting underutilized network devices or links to sleep. If a network device has a few links such as a backbone network device, it is better to consider underutilized device to put to sleep, but if a network device has many links such as in data centers network devices and campus networks devices, it is better to consider underutilized links to put to sleep. In this paper, we consider putting underutilized links to sleep. Off course, if all links of a network device are put to sleep, the device itself will be put to sleep. Furthermore, in this paper we consider OpenFlow-based network as it is software defined network and flexible to control network devices. Besides, OpenFlow is being standardized by big enterprises such as NEC, Google, Cisco, Microsoft etc. We evaluated our approach by simulation and compare it with node based approach to see its effectiveness.

Keywords: *OpenFlow network, energy efficient network, link utilization*

1. Introduction

There are a lot of researches on energy efficient wireless networks, especially on wireless sensor networks and mobile ad hoc networks. As nodes in these networks are operated by power constrained batteries, the aim of the researches is to extend the network life by reducing the energy consumption. However, recently researchers and industries have started putting their attentions on energy efficient wired networks also [1]. Unlike energy efficient wireless networks where the motivation is to extend the network's life, the motivation to develop energy efficient wired network, where the power supply is considered to be infinite, is to actually save energy without degrading the performance of the network.

There are mainly two reasons why wired network operators have interest in energy aware networks. One is due to the environmental awareness. There is strong political and social pressure to industries to develop energy efficient systems and networks. Another is that due to Internet traffic increase in recent years, the power consumed by networks has also increased. The Global e-Sustainability Initiative (GeSI) [2] estimated that in 2010, network energy requirement in European telecom operators was about 21.4 TWh and forecasted a figure of 35.8 TWh in 2020 if no power saving initiatives are adopted.

In wired networks, for robustness and fault tolerance, over provisioning of link bandwidth and massive link redundancies are adopted. It is stated in [3] that the average utilization of backbone networks/links is less than 30 percent. The experiment has found that the energy consumption by a network equipment is almost the same whether it is idle or fully utilized [4]. The conclusion is, a network equipment's power consumption is independent of its utilization. From the findings of above works, we see that a large portion of energy is consumed by redundant links and idle or underutilized network devices. Observations have shown that, over a large timescale, the Internet traffic exhibits strong daily and weekly patterns which do not change over years [5]. Moreover, it has been shown in [6] that higher the link speed, more energy consumption by the device/link. They have shown that a 1 Gbps Ethernet link consumes about 4W more than a 100 Mbps link.

Taking the advantages of the aforementioned characteristics of wired networks—underutilization of network devices, regular network traffic patterns and lower the link speed lower the power consumption—researchers have proposed different methods to put some links of a device or the device itself to sleep or lower link speed depending upon the traffic volume in the network. The basic approach is to measure traffic at each device and depending upon the traffic volume, actions for reducing the power consumption, such as shutting down some links/devices or reducing link speed are taken such that the overall network performance does not degrade.

In this paper, we propose a method to shutdown (put to sleep) underutilized links of routers/switches (hereafter referred as nodes), in OpenFlow based networks [7]. If all links of a node are underutilized the node itself is put to sleep. Similarly, when the utilization of remaining links, which are alive, passes certain threshold, the sleeping links of the node are awakened. In order to measure the utilization of network links, the controller in an OpenFlow network periodically collects network traffic passing through links of each node. Based on the traffic flow in a given period of time, the controller calculates the utilization of each link of each node in the network. By putting the underutilized links to sleep, power consumption in the network can be reduced significantly.

The paper is organized as follows. In Section 2, we explain the related works followed by a brief description of OpenFlow architecture in Section 3. In Section 4, we give the detail of our proposed approach to reduce power consumption in OpenFlow based networks. In Section 5, we show the performance evaluation of our approach followed by conclusion in Section 6.

2. Related Works

Gupta *et al.* [8, 9, 10] have first mentioned the energy efficiency issue in wired networks. They have developed different algorithms to find inactive period of links and put them to low power mode during that period. They have not mentioned how long to turn off the links. Without detailed knowledge of the incoming traffic pattern, this approach has to compromise with network performance on energy consumption reduction. Moreover, this approach cannot be applied to backbone routers. An adaptive link rate (ALR) for Ethernet is proposed in [4]. When a link is underutilized, the speed of the link is reduced to save energy. However, the proposed method only works for Ethernet. More research is needed to extend it to other network links. Arai *et al.* [11] have proposed a distributed routing protocol for the network running OSPF (Open Shortest Path First) [12] routing protocol to put some nodes to sleep. Besides, OSPFs LSA (Link State Advertisement), and LSDB (Link State Database), their proposed protocol, floods NSA (Network State Advertisement) and maintains NSDB

(Network State Database) and HLDB (Historical Link State Database). As the original OSPF cannot distinguish between sleeping and failure nodes, HLDB is used to distinguish such nodes in their proposed method. The routers calculate traffic trend from NSDB and change weight of links and then apply standard OSPF operation. Backbone nodes, which are not in the path of any one pair of access nodes, shift to sleep mode. If the traffic increases the sleeping nodes are put to awake. The main problem of the proposal is that routers have to maintain three large databases (LSDB, NSDB and HLDB). There will be large control packets flow in the network as the routers have to flood both NSA and LSA regularly. Computation to shift nodes to sleep and vice versa is also high for routers.

In [13], a scheme to put a node to sleep depending upon the network traffic in OpenFlow network is proposed. From the network traffic flowing through the network, which is collected by the controller, and the network capacity, which is already known from nodes' specification, the number of required nodes to handle the traffic is calculated. If the required number of nodes is less than the alive nodes in the network, an access tree, with the node handling the highest traffic volume as a root node, is constructed. The leaf nodes which are not directly connected to hosts are put to sleep until the required number of nodes is the same as the alive nodes. The method considers homogeneous network devices and works well when all links of a node can be put to sleep. If some links of a node are underutilized but some are not, the node will not be put to sleep.

In this paper, we consider utilization of each link of a node in the network instead of the required number of nodes only as in [13]. If a link's utilization is below sleep threshold, it is put to sleep. If alive links' utilization of a node is above wake up threshold, the node's sleeping links will be put to awake. If all links of a node are underutilized, the node will be put to sleep. However, links or nodes that are directly connected to hosts (PCs/servers) or partition the network will not be put to sleep. Our link based approach fine tunes the node based approach proposed in [13].

3. OpenFlow Architecture

In a classical network device, such as a router and a switch, the controller, *e.g.*, routing protocols and algorithms, and forwarding tables which are generated by the controller to forward packets reside on the same device. Users could not easily implement their protocols and algorithms in these devices. Only vendors of the devices could upgrade and implement the protocols and algorithms. In short, the network devices are not flexible for researchers for their research and development.

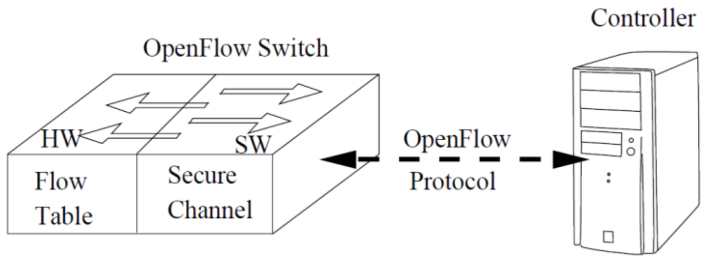

Figure 1. OpenFlow architecture

To overcome these limitations, in OpenFlow the controller and forwarding tables are separated [14, 7]. The forwarding table or data path still resides on routers/switches, while high-level routing decisions are moved to a separate controller, typically a standard server as shown in Figure 1. The OpenFlow switch and the controller communicate via the OpenFlow protocol using secured channel. The OpenFlow protocol defines messages such as packet-received, send-packetout, modify-forwarding-table, get-stats and so on. Basically, the controller configures routers/switches, manages flow table (routing) based on the network state and routers/switches state. Users can implement their own protocols and algorithms to control network devices. Due to the flexibility of the OpenFlow based network, it is being standardized and supported by Open Networking Foundation whose members are Google, Cisco, Microsoft, facebook, *etc.* and is considered to be the future network architecture for flexibility and maintainability [15].

4. Proposed Energy Saving Approach in OpenFlow Based Networks

In this section, we explain the assumed network environment, link utilization and proposed link sleep and wake up procedures with an example.

4.1. Network Environment

Our proposed method is for OpenFlow based network which consists of a controller and a number of nodes (switches/routers) as shown in Figure 2. There are two types of nodes; access nodes to which users terminals are connected and backbone nodes which are intermediate nodes residing between access nodes. The controller communicates with each node using secure channel and OpenFlow protocol. Among others, it collects network statistics and updates flow-table of each node. For simplicity, we consider that all links have the same link speed and power consumption and also each node is capable of recording the traffic passed through its links.

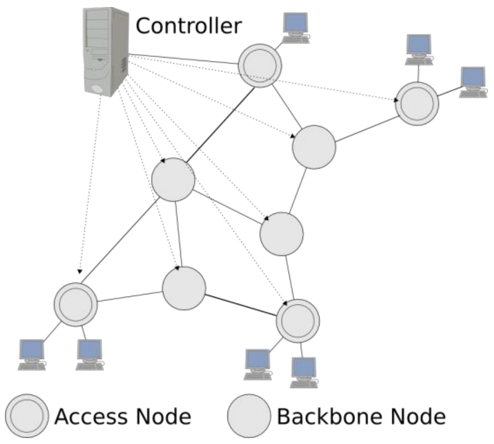

Figure 2. OpenFlow Based Network

4.2. Link Utilization

The utilization of a link *i* is calculated using the following formula.

$$U_i = D_i/(C_i \times t) \qquad (1)$$

where

U_i is the utilization of link *i*.

D_i is the total data traffic passed through the link *i* in time *t*.

C_i is the capacity/bandwidth of the link *i*.

t is the time interval to measure link utilization.

The administrator can set the time interval to suit the network condition but it should start after the controller updates flow table and put links to sleep or wake up.

Link sleep threshold: It is a value to decide if a link will be a candidate to sleep or not. If a link's utilization is below the sleep threshold, the link is a candidate for sleep. Though a link may be a candidate for sleep, it will not be put to sleep if a host is connected to it or it will partition the network if it is put to sleep. Setting up a threshold value is difficult as it will depend upon the network traffic. However, with experiment, it can be set at a desired value.

Link wake up threshold: It is a value to decide if sleeping links/nodes will be put to awake. If a node has both sleeping links and alive (awake) links and if one or more of alive link's utilization is above the link wake up threshold, then all sleeping links of the node will be put to awake instead of waking up links one at a time because it will need a complex decision making procedures based on other network factors.

4.3. Link Sleep and Wake up Procedure

The link sleep and wake up procedure is as stated below. It is obvious that the links of access nodes directly connected to hosts (PCs/servers) will not be put to sleep. However, links of access nodes which are connected to other nodes can be put to sleep if they do not partition the network.

1) The controller sends request to each node under its control to send the amount of traffic passed through each link.
2) Each node replies with the amount of traffic passed through its links.
3) The controller calculates the utilization of each link and identifies two types of link: *Type i)* links whose utilization is below the link sleep threshold.

   Type ii) links whose utilization is above the link wake up threshold.
4) If only *Type i)* links exist then step *4A* is performed.

   If only *Type ii)* links exist then step *4B* is performed.

   If both *Type i)* and *Type ii)* links exist then step *4C* is performed.

   4A.

   i). The controller checks which links are eligible to go to sleep, *i.e.*, utilization is below the link sleep threshold and they will not partition the network and are not connected to hosts.

   ii). The controller calculates flow tables (paths) from each access node to other access nodes (For simplicity, we consider shortest path first here.) without considering sleep eligible links.

   iii). The controller forwards the table to nodes (where the flow table has changed) in the network. After receiving the table, nodes use the new flow table to forward packets.

   iv). The controller then sends request to nodes with sleep eligible links to put the links to sleep. If all links of a node are eligible to sleep then a request to go to sleep will be sent to the node.

4B.

i). The controller checks if the node with *Type ii)* links have links that were put to sleep. If there are, the controller sends request to wake up the sleeping links. As a result in some cases the node may be requested to wake up if it was put to sleep.

ii). The controller calculates the path from each access node to other access nodes including woken up links.

iii). The controller sends new flow tables to concerned nodes which will start forwarding the packets using the new flow tables.

4C.

i). First the controller wakes of sleeping links as shown in 4B.

ii). The controller calculates the path from each access node to other access nodes including woken up links and excluding sleep eligible links.

iii). The controller sends the flow tables to concerned nodes which will start forwarding packets with the new flow tables.

iv). The controller sends request to concerned nodes to put the links to sleep.

5) Repeat the process in set regular interval.

The controller will not put links which are connected to hosts/servers (thus access nodes will never be put to sleep) and which causes network to partition if links are put to sleep

4.4. Example

Let's assume a network as shown in Figure 3(a). The link DE will never be put to sleep as it will partition the network. Let's assume that in Figure 3(a), link utilization of links BF and BG is below the link sleep threshold, *i.e.*, *Type i)* links. The procedure 4A is executed.

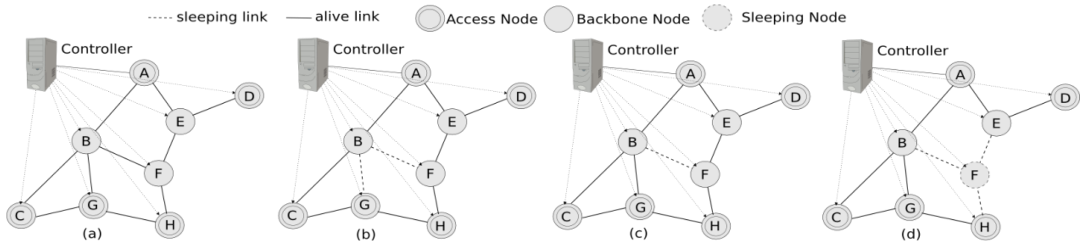

Figure 3. Example of link sleep/wake procedure

Using step *4A.* i), the controller checks if the network will partition when links BF and BG are put to sleep. Here it will not.

Using step *4A.* ii), the controller calculates flow tables (paths) from each access node to other access nodes excluding links BF and BG. For example, paths from node A to other access nodes are A-E-D, A-E-F-H, A-B-C and A-B-C-G.

Using step *4A.* iii), the controller sends flow tables to each node whose table has changed and the node starts to use the new table.

Using step *4A*. iv), the controller sends request to nodes B, F and G to put links BF and BG (*i.e.*, respective ports) to sleep.

After the procedure *4A* is completed, the network topology changes to as shown in Figure 3(b).

Now, suppose link utilization of the link CG in network shown in Figure 3(b) is above the link wake up threshold, *i.e.*, *Type ii)* link. The procedure 4B is executed.

Using step *4B*. i) the controller finds that node G has a link that is sleeping and sends wake up request to nodes G and B to wake up the link BG.

Using step 4B. ii), the controller calculates flow tables from each access node to other access nodes including the woken up link BG. For example from node A to other access nodes, paths will be A-E-D, A-E-F-H, A-B-C and A-B-G.

Using step *4B*. iii), the controller sends new flow tables to nodes which will update their existing tables. After the procedure *4B* is completed, the network topology changes to as shown in Figure 3(c).

In next example, suppose link utilization of links EF and FH in network Figure 3(c) is below the link sleep threshold, *i.e.*, *Type i)* links. After completing the procedure *4A*, links EF and FH are put to sleep. Since all links of node F are in sleep, the controller sends sleep request to node F to go to sleep. The network topology changes to as shown in Figure 3(d) in which not only links are put to sleep but a node with all links in sleep is also put to sleep.

It is important to note that if a sleeping link of a node is needed to wake up, all of its sleeping links will be put to awake also because it is difficult to decide which link we need to wake up without analyzing the properties of the traffic.

5. Performance Evaluation

We perform simulation to evaluate our proposal using virtual OpenFlow network.We used NOX [16] as the controller software and mininet [17] to construct virtual OpenFlow network. In our simulation we constructed 5 network topologies each consisting of 15 access nodes and 30 backbone nodes placed randomly in the network. One of the topologies is as shown in Figure 4. For each topology, the simulation is performed ten times and each time for ten minutes. To calculate the link utilization, 15 seconds network traffic is considered. We used the link sleep threshold as 10% and the link wake up threshold as 30%, i.e. if a link’s utilization is less than 10% it is eligible to go to sleep and if it is more than 30% then the node having this link needs to wake up its sleeping links. We compare the following properties with the node based approach [13].

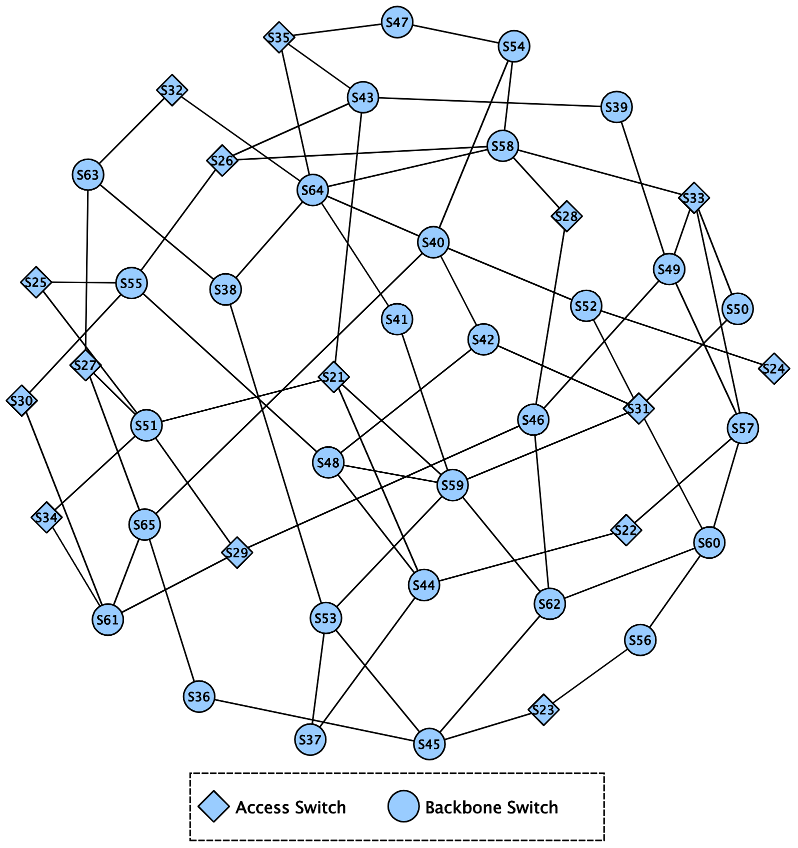

Figure 4. One of the simulated topologies

1) Number of eligible sleeping links.

2) Power consumed by alive links in the network.

3) Average processing time.

5.1. Number of eligible sleeping links

From the simulation results shown in Figure 5, we see that more links are eligible to sleep in the link based approach than the node based approach. It should be noted that in the node based approach, all links of a node should be eligible to sleep before it can be put to sleep otherwise its links are not eligible to sleep. In the link based approach, individual link is considered thus a node may not be eligible to sleep but some of its links may be eligible to sleep. Thus the link based approach shows the clear advantages over the node based approach. Moreover, if all links of a node are eligible to sleep then the link based approach will be the same as the node based approach.

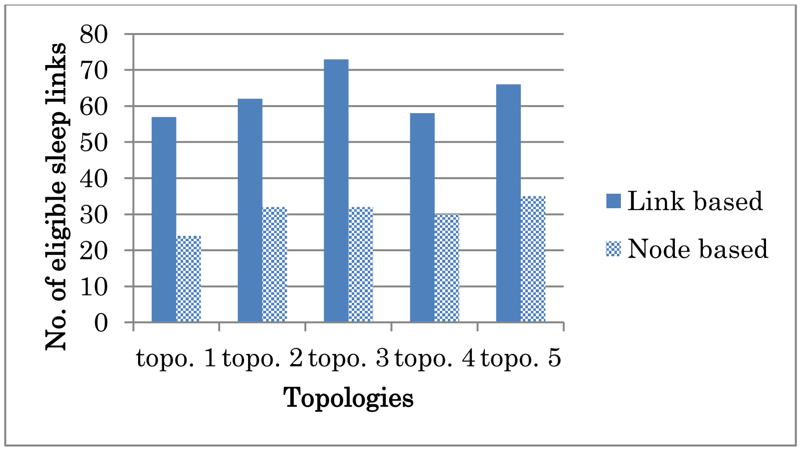

Figure 5. Eligible sleeping links

5.2. Power consumed by alive links in the network

A node's power consumption is expressed as shown in the following equation [18].

$$P_T = P_f + P_p \times n \qquad (2)$$

where

P_T is total power consumed by the node

P_f is node's basic power consumption (motherboard, cooling system etc.).

P_p is the power consumed by a link port (link).

n is the number of alive link ports.

In our simulation we set $P_f = 3000(\mathrm{W})$ and $P_p = 3(\mathrm{W})$.

After links/nodes are put to sleep we calculate the power consumed by the remaining alive links of nodes. From the result shown in Figure 6, we see that the link based approach uses less power than the node based approach. In the link based proposal, even though a node is alive, some of its links may be put to sleep whereas in the previous approach if a node is alive all of its links are alive and thus the power consumption is more.

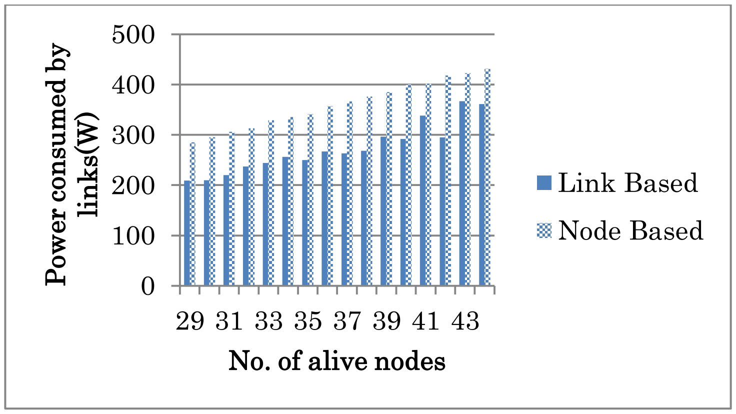

Figure 6. Power consumed by alive links

5.3. Average processing time

We calculated the average time required to find eligible sleep links, put links/nodes to sleep or wake them up and change the flow table. As shown in Figure 7, we found that the link based approach needs about two times more processing time than the node based approach. In the previous node based approach, each node is checked one at a time whereas in the link based approach, each link is checked. As in any network, since there are more links than nodes, the processing time in the linked based approach is higher than the node based approach.

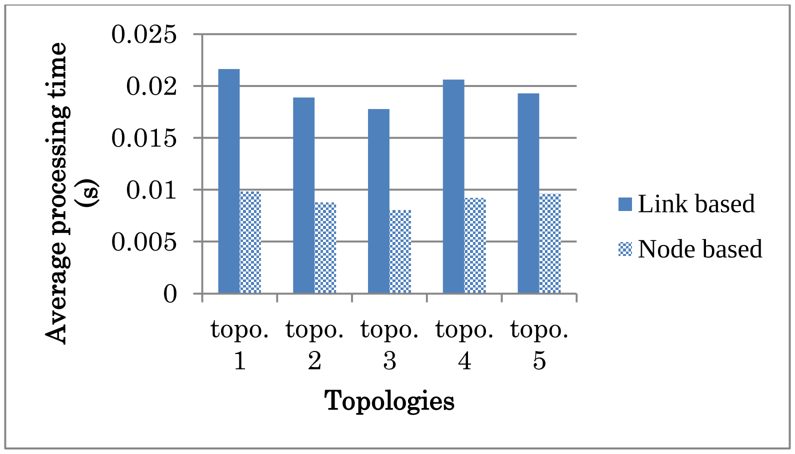

Figure 7. Average processing time

6. Conclusions

In this paper, we proposed a link based approach to put underutilized links to sleep to save power consumed by OpenFlow based networks. It has a clear advantage over the previous node based approach where a node is put to sleep if all of its links are eligible to sleep. We have considered links utilization only to decide if links/nodes can be put to sleep or not. We need to consider network traffic characteristics also. For example, if a link/node is put to sleep, will there be any increase in packet arrival delay. In future, we will investigate packet delay, packet loss, in short, impact to end-to-end data packets due to putting links/nodes to sleep.

References

[1] R. Bolla, R. Bruschi, F. Davoli and F. Cucchietti, “Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures”, Communications Surveys & Tutorials, IEEE, vol. 13, no. 2, **(2011)** Second Quarter, pp. 223–244.

[2] Global e-Sustainability Initiative (GeSI), “SMART 2020: Enabling the low carbon economy in the information age,” Global e-Sustainability Initiative (GeSI), SMART 2020, http://www.smart2020.org/ assets/files/02 Smart2020Report.pdf, **(2008)**.

[3] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy and D. Wetherall, “Reducing network energy consumption via sleeping and rateadaptation”, Proceeding NSDI’08 Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, **(2008)**, pp. 323–336.

[4] C. Gunaratne, K. Christensen, B. Nordman and S. Suen, “Reducing the energy consumption of ethernet with adaptive link rate (ALR)”, IEEE TRANSACTIONS ON COMPUTERS, vol. 57, no. 4, **(2008)** April, pp. 448–461.

[5] C. Fraleign, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. ROcell, T. Seely and S. C. Diot, "Packet-level traffic measurements from the sprint IP backbone", IEEE Network, vol. 17, no. 6, **(2003)** November-December, pp. 6–16.
[6] C. Gunaratne and K. Christensen, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed", INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT, vol. 15, no. 5, **(2005)** September/October, pp. 297–310.
[7] OSS, "OpenFlow switch specification", http://www.openflow.org/wp/documents/.
[8] M. Gupta and S. Singh, "Dynamic ethernet link shutdown for energy conservation on ethernet links", IEEE ICC 2007, **(2007)** June, pp. 6151–6161.
[9] M. Gupta, S. Grover and S. Singh, "A feasibility study for power management in LAN switches", Proc. IEEE ICNP 2004, **(2004)**, pp. 361–371.
[10] M. Gupta and S. Singh, "Using low-power modes for energy conservation in ethernet lans", INFOCOM 2007, **(2007)**, pp. 2451–2455.
[11] D. Arai and K. Yoshihara, "Eco-friendly distributed routing protocol for reducing network energy consumption", Proc. of CNSM 2010, **(2010)** October, pp. 104–111.
[12] RFC, "2328 OSPF version 2", **(1998)**.
[13] B. B. Bista, M. Takanohashi, T. Takata and D. B. Rawat, "A power saving scheme for OpenFlow network", Journal of Clean Energy Technologies, vol. 1, no. 4, **(2013)**, pp. 276-280.
[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: Enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, **(2008)**, pp. 69–74.
[15] ONF, "Open networking foundation", https://www.opennetworking.org/.
[16] NOX, http://noxrepo.org/wp/.
[17] Mininet, "An instant virtual network on your laptop (or other pc)", http://mininet.github.com/.
[18] G. Ananthanarayanan and R. H. Katz, "Greening the switch", Proceedings of the 2008 conference on Power aware computing and systems, **(2008)**.

Authors

Bhed Bahadur Bista received his Ph.D. degree in Information Science from Tohoku University, Japan in 1997. He is currently working as an Associate Professor at Iwate Prefectural University, Japan. His research interests include energy efficient networks, mobile and wireless networks.

Arata Fukushi graduated from Iwate Prefectural University, Japan, in 2013. Currently he is working in a company in Japan. He is interested in power aware computer networks. The above research was done while he was in Iwate Prefectural University.

Toyoo Takata received his Ph.D. degree in information and computer sciences from Osaka University, Japan, in 1989. Since 1998, he has been a Professor of Software and Information Science, Iwate Prefectural University. His research interests include network security, user authentication and information theory.

Danda B. Rawat is currently working as an Assistant Professor at Georgia Southern University, USA. He received his Ph.D. in Electrical and Computer Engineering from Old Dominion University, USA. His research interests include areas of wireless communications and network, and network security.